n. /SWITCH/ means "put up slide number n."

Put up the Title Slide

2. /SWITCH/

I like to think about how artists and scientists work. I also like essays. An essay is a written attempt to understand something. This presentation is an essay—there are not many worked-out lessons here. In this sense, this is an artistic talk.

3. /SWITCH/

Prefrontal lightningbolt too lazy to chew the sphinx's loudest eyelash
Not even if it shushes you with a mast of sneers
Down which grateful bankvault-doors scamper
Because of a doublejointedness that glows in the dark
Like a soliloquy of walnuts
Numbed by beaks of headless measuringtape
So the lubriciousness can tower in peace
Like a buzzsaw trapped in a perfumery of shrugs
Lemon
Or lime
Only a maze can remember your hair of buttered blowguns

▨ ▨ ▨

From *Nights of Naomi* by Bill Knott.

Art is strange. Art cannot be understood. Robert Browning is reported to have said of a passage he wrote:

> *"When I wrote that, God and I knew what it meant, but now God alone knows."*

Science and art don't seem directly related.
Except people do science.
And people do art.
Science is a clear statement of truth in the actual world.

4. /SWITCH/

Behold the simplicity of science.
This is a sort of contemporary science which is more easily understood than the nonsensical poem.
Don't you think?
/PAUSE/
Ah, but with training this is quite easily understood. The poem, however—obviously no amount of training, teaching, or learning will bring you up to understanding it.
Some other examples.

5. /SWITCH/

A visual representation of source code shows, clearly, the structure of the program, whereas this Jackson Pollock painting is clearly less understandable

6. /SWITCH/

Some mathematicians have computed the fractal dimension of some of Pollock's drip paintings as high as 1.72—thickly layered and complex.

7. /SWITCH/

Other representations of source code are more prosaic, and tell the story of a human mind at work using human hands, and sometimes I wonder whether writing code out by hand helps the programmer better understand what he or she is doing.

8. /SWITCH/

Some human designs seem a bit sloppy, or even pieced together, but still we can see the human mind at work even through the mess and erasures.

9. /SWITCH/

This work is based on a recursive writing / erasing / rewriting process in which a poem by Wallace Stevens is written, erased, and then the same poem with one fewer line is written again; the process halts when only one line—the first line—remains. A surrounding recursion produces a total of 24 pieces like this where the first is the most heavily written and erased and the last is the least. This is the 13th in the sequence.

10. /SWITCH/

Kevin Sullivan of the University of Virginia, working with Sanda, captured her process in Java code, and this is the heart of the code and process.

Art is, in part, the practice of intense and disciplined noticing—of details, of life, of how the world works—and in that way at least it resembles science.

Many artists use processes or formal techniques to increase the opportunities of noticing—artists don't simply visit their cafés every day, do their laundry, and hope to see stuff to put in their art. Some of those techniques include types of sampling, stochastic or otherwise.

11. /SWITCH/

For many, science is a caricature—a process and a way of thinking; individuals practice that process and that way of thinking to perfection, the result being a progression of increased knowledge over the millennia—the forward march of science.

In fact, this is not the case. Not only is it not clear what science really is and how it's practiced, but in terms of purely being right, science is one of the most failure-prone of disciplines, with theories being overturned all the time. Aristotelian science overturned by Newton overturned by Einstein, Ptolemy overturned by Copernicus, Lamarck by Darwin. And on and on.

12. /SWITCH/

Along with that caricature is a companion for engineering. Engineering, it's thought, is the application of science, and with that is a belief—perhaps not deeply held but at least a first approximation—that science precedes engineering, and that without science there could be no engineering.

Important in thinking about art, science, and engineering are the ideas of *explicit* and *tacit* knowledge.

Explicit knowledge is something that can be written down and is thereby fully and accurately transmitted. Here is an example:

Los Angeles is east of Reno.

For some this can be new information, and perhaps it requires some verification, but once you have read, understood, and believed this sentence, you know it as well as anyone.

Tacit knowledge depends on practice and, to an extent, skill. Knowing how to design, for example, requires tacit knowledge.

13. /SWITCH/

Artists are thought to use a trained sense of aesthetics to work effectively. Artists typically have a highly trained skill or talent, like the ability to draw, paint, sculpt, write, or compose music.

When viewed according to these caricatures, art, engineering, and science appear to be very different disciplines requiring different abilities, skills, and likes.

14. /SWITCH/

Science is a complicated beast. People who study how science works cannot agree on what they see. For a few minutes I'll give you a quick tour of the landscape, each the view of a respected philosopher of science.

15. /SWITCH/

The study of what science is focuses on the question of what separates scientific ideas, statements, and theories from pseudoscience—or voodoo, if you will. This is called the demarcation problem.

Further, the demarcation problem includes finding out how "science" determines whether one theory is "better" than another, thereby invoking a scientific revolution in the Kuhnian sense.

16. /SWITCH/

Positivism has been the dominant "theory" of science for many centuries, and particularly so in the 20th and 21st centuries. In fact, it is the current received theory. It rejects any appeal to metaphysics, and can be thought of as the basis of rejection of creationism and other religious theories of the universe. Positivism states that we understand the world by looking at it—that reality is real, and that our sense perceptions are valid. This holds—positivism assumes—even when our perception is through mechanisms we create, such as scientific instruments.

Logical positivism adds the idea that rational or logical conclusions from observations and other scientific conclusions are also valid pieces of knowledge about the world.

17. /SWITCH/

Science proceeds by scientific publication. What counts as science is what's published in conferences and journals that are labeled "scientific."

Such publication is based on the opinions of a jury of peers—in computer science we call it *peer review*. Such juries are not elected or randomly selected but are chosen by program chairs and editors in chief, and so represent a kind of scientific *in group*, an inner circle.

18. /SWITCH/

Inductivism tackles the problem of how to make universal statements given a finite and limited set of observations. It's like naïve empiricism, but permits many similar observations to add up to a general statement about the universe. Inductivism is how we come to believe that a series of observations on or near Earth lead to valid scientific theories about the other side of the galaxy, other galaxies, and in fact, the other side of the universe.

It would be a statement of metaphysics to claim that our neighborhood of the universe is special, with laws that enable the existence of cognitive beings, for example. But to claim that our neighborhood is *not* special is science, according to inductivism.

19. /SWITCH/

Probabilism is a brand of inductivism. Scientific statements are ones that can be shown to be probable, and one theory is better than another if it is more probable. Voodoo is not scientific because its probability is low—or perhaps it's simply a discredited scientific theory because most other theories are more probable.

The size of the universe and the small sample we have of it naturally mean that the probabilities of our theories—if they are ever actually computed—are small.

20. /SWITCH/

Conventionalism kind of ducks the whole question of what's true in order to get at what's useful. In a sense, conventionalism takes engineering more seriously than it does truth, and posits that science is about coming up with mechanisms that make engineering calculations easy—and perhaps elegant.

21. /SWITCH/

Falsification is considered by many to the be primary characteristic of a scientific statement. If a statement can be falsified, it is scientific. Because an experiment can be an observation, there are many statements that are scientific that we might not think of that way.

2+2=3 is a scientific statement that happens to be false (by observation and elementary-school arithmetic).

This sentence no verb is similarly scientific, and true. Let me say that sentence again: *This sentence no verb*.

More importantly, there are two questions that make things problematic. The first is that if you need to construct an instrument as part of an experiment to verify a statement, and that instrument and understanding of what it measures depends on the theory being tested, then why should we believe that the instrument is capable of falsifying the theory?

22. /SWITCH/

The other is that if a theory is well-established and an experiment falsifies the theory, then the question arises about how seriously to take the refutation. In some cases the experimental data is thrown out. If it persists, usually the theory is patched to account for the discrepancy.

And sometimes a deep analysis is undertaken to determine whether some variables are disturbing the results (the *ceteris paraibus* (*kay-ter-is par-ib-us*) or "all things being equal" situation).

23. /SWITCH/

Imré Lakatos tried to reconcile Popper's rational falsificationism with Thomas Kuhn's historical view of science. Popper required instances of nature saying **NO** be explained or the theory replaced; Kuhn observed "normal science" moving forward patching its theory until it was too unwieldy, and then replacing it during a sort of revolution.

Lakatos claimed that science was performed in the form of a *research programme* in which a *hard core* of the theory was protected from direct patching by a cocoon of auxiliary hypotheses that could be modified to handle experimental discrepancies. Research programmes contain methodological statements that tell scientists what is good to work on and what isn't.

When a new research programme comes along that is better or "progressive," it might replace the older one.

24. /SWITCH/

Paul Feyerabend—in some ways my favorite philosopher of science—said that science really doesn't proceed in a rational manner, that there is no such thing as demarcation between science and voodoo, that there is no such thing as progress, and that one theory succeeds another only when the propaganda machine of the replacing theory's proponents overthrows the other's.

25. /SWITCH/

Bruno Latour looks at science with a sociologist's eye. He argues three ideas: science is what a jury decides it is; science is what gets done in a laboratory; accepted science is the result of a strong social network in which references accumulate to indicate what the overall community of scientists thinks about the work.

In other words, to know what science is, visit Google Scholar.

26. /SWITCH/

When I first started reading how philosophers and historians looked at science and how it was really done, I was shocked. You mean science is not purely about the truth, that scientists weren't objectively securing the truth from the clutches of nature, that science might after all simply be a convenient story about the world that enables us to build things?

And what of the idea that engineering depends on science? Here is how the *American Engineers' Council for Professional Development* defines engineering:

> *The creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes....*

27. /SWITCH/

In 1995 I went back to school and got my MFA in Creative Writing—in poetry in fact. They tried to teach me how to notice things. In fact, the primary method of teaching fine arts is to show you how to observe and analyze closely so you can learn and improve on your own. If all you have for a teacher, for example, is a complicated poem written by a master, you must be able to see all the craft elements and thought processes that must have gone on in the poet to create that poem.

We think of engineers as being like craftsmen or even laborers, no matter what we observe.

28. /SWITCH/

But building things—as engineers do—requires people to manipulate materials and the physical world, and figuring out how to build things requires looking at the world very carefully. Engineers need tacit knowledge to be able to do their work well. And tacit knowledge comes from practice and discipline, and being able to notice deeply.

29. /SWITCH/

It shouldn't be surprising then to notice, when we look back on the history of science, that many or perhaps most of our scientific advances came from engineers, builders, and craftsmen working with materials, coming to understand them, and formalizing their characteristics.

What science contributes, mainly, is a way of thinking about materials and forces in ways that are computationally convenient and repeatable. Science is a story with lessons that can be easily applied. These stories are in a language that is computationally friendly—called mathematics.

30. /SWITCH/

One good example is the steam engine. It was developed while scientists were way into the caloric theory of heat. Looked at now, this theory is hilarious. It held that heat consisted of a fluid called *caloric* that flows from hotter to colder bodies. Caloric is a weightless gas that could pass in and out of pores in solids and liquids.

The amount of caloric in the universe is constant, so there is a sort of conservation that thermodynamics also recognizes.

A cup of hot coffee cools because caloric is self-repelling, so it disperses from the dense concentration in the cup to the less dense air.

When caloric enters the air, it combines with its molecules, and thus the air increases in volume. By expanding on what happens when caloric interacts with other matter we can explain heat radiation, phase changes (ice, water, steam, for example), as well as deduce most of the gas laws.

When Laplace corrected Newton's pulse equation with a constant to take caloric into account, the equation was better able to predict the speed of sound, and as this constant was refined under the caloric theory, even more precise predictions were made for over a century.

The caloric theory was replaced, generally, by thermodynamics, but you know, the whole time steam engines continued to work, and they still do.

31. /SWITCH/

Here is a more realistic view of scientists and engineers. Both scientists and engineers create knowledge, are problem driven, and seek to understand and explain the world and things in it. Scientists design experiments and engineers design devices—in fact, engineers help scientists design the devices used in their experiments. Scientists are more comfortable with abstract ideas, and engineers with concrete or contingent knowledge. And both rely on tacit knowledge.

It's possible to become a scientist by reading books and the literature, but it's a lot easier by getting a PhD where you are mentored by a working scientist, the same way an artist is.

32. /SWITCH/

Discovery is at the center of scientific inquiry. Science can be broken into two parts: discovery and verification. Verification is mostly what the philosophers of science concern themselves with, but without discovery, there is not so much to verify. Engineering, as we've seen, is one way discoveries happen—from playing with materials, trying to bend them to our wills, using them to build with—and another way is simple noticing what's in front of you.

33. /SWITCH/

Artists discover; it's what they're trained to do. It requires some talent to become an artist, but mainly it takes practice and perseverance. Genius is rarely if ever part of the equation.

34. /SWITCH/

Exploration and discovery are the heart of any artistic endeavor. Noticing takes place while on a journey of some sort. Artists often start out with no particular destination in mind, and sometimes—this will be hard to believe at first—no starting point in mind but simply what happens to occur to them at the initiation of the process.

Some artists use artificial mechanisms to ensure that the starting place and subject matter are substantially random and unplanned for.

During these early stages it's important to have a sort of defocussed attention and to engage in flat associations—going broad not deep. We can call this exploration, but by noticing we begin to discover. Discovery is making connections, and this is where metaphors pop up.

Some writers I know start with a scene or a situation, sometimes just a sense or feeling—and that becomes a novel. The poet Bob Hass says he starts with a phrase, a word, or a line and a gesture, a physical gesture in the air which indicates, for example, a rising energy level or a decline or some more complicated shape—and the result is a poem.

35. /SWITCH/

Once a draft begins to take shape—a draft of a written piece or paint on the canvas or some shards off the rock or some notes on the guitar—a process of understanding begins. This is the revision process where what has been discovered is examined and the best story / painting / sculpture / composition is created. Sometimes these drafts are in the form of sketches or studies when the medium is not malleable.

This is like the full process of science—exploration, discovery, and then verification or understanding. What's different is the emphasis given to the parts. Serious thinkers about the scientific process focus on verification and understanding. Making judgments about what's good science. What's nice about art is that there has been plenty of thought given to the first parts of the process: exploration and discovery.

36. /SWITCH/

Artistic creation is as much about being lost as exploring known—even half-known—territory. Great explorers don't explore the parts of the map that have names. And, as with earthly explorers, you sometimes find yourself in someone else's home.

Creation's companions, then, include despair, anxiety, self-doubt, and pointlessness. We don't hear about these so much from scientists or philosophers of science.

37. /SWITCH/

And there's a nagging issue: at the beginning of recorded history there is a set of stories about how and why the world and people were created, why weather happens, why the earth moves under our feet, why the seasons change, why people are different, why their moods change, why there are families, how to plant, hunt, and live. Oceans, sky, fire, trees, animals.

We call these stories literature now, or religious tracts. But at the time, they passed as science.

38. /SWITCH/

The best art typically comes with struggle and as a surprise. Not always, but in my case it's usually news which of my works is considered my most important.

39. /SWITCH/

Looking at art objectively, given this more enlightened view, it seems that artists, engineers, and scientists are not so different after all.

They all create knowledge of one sort or another—sometimes "hard science" knowledge, sometimes "how to" knowledge, and sometimes "what about it" knowledge. Artists create language and other sorts of realities (like engineers). They work through the medium of the pieces of art they endeavor to create. Theses pieces are devices used to understand and explain the world. In this they are like scientists who build instruments. And they work with what's in front of them, using skills and talents that take time to acquire.

40. /SWITCH/

Ok, I know what you're thinking. You're thinking: "gee, this is all really, really interesting, and he seems like a smart and broadly informed guy, but this is a product line conference, and I was hoping to hear something about that."

So right now we're going to take a little break from the main path of the talk and detour into my experience with product lines, and maybe what we did way back when will be a little different from what you're used to, and might even inform you a tad about what might be possible from the point of view of an artist.

41. /SWITCH/

Back in 1984 I started a company to produce commercial implementations of Common Lisp, which at the time was thought of as an artificial intelligence language.

In 1984 the closest we came to a usable "personal computer" was the workstation, which was a $20,000–$100,000 affair. And a Lisp implementation was like a language system and operating system combined. In addition to a compiler, there was an extensive runtime system that handled runtime type checking, a threading / scheduling facility, a complex memory manager including a pseudo-realtime "ephemeral" garbage collector, foreign function interfaces to a variety of programming languages, an extensive programming environment including a text editor, a debugger, a data inspector, a code visualizer, and a graphics / window system—sometimes needing to extend all the way down to bare metal.

Naturally, the compiler is usable from user-code (a program could produce Lisp code as a data structure, compile it, and run it). This means there is an interpreter and a command-line type interface, so it really felt like you were talking to an operating system with a built-in language system as part of it.

We supported this language implementation—equivalent to the Symbolics Lisp machines—on 30 different platforms, each of which was an CPU / operating system pair.

As you might guess, we needed to invent the idea of a software product line.

42. /SWITCH/

We called what we were doing "portability," but we wanted a large common code base shared among all the platforms, and a small code base for each one. And we wanted the shared code to run really well on each platform, because at the time, performance was the name of the game.

We were an OEM company, so we took contracts to port our system to the customer's platform or platforms, and we were paid a fee for the port plus a royalty stream. It was called a "private label" business model.

The portability problem was broken into two parts: how to partition the compiler and code-preparation part of the system into a large common code base with a small code base for each platform, and how to partition the runtime system into a large shared code base for all platforms and a small code base for the connections to the hardware and operating system.

The latter was relatively simple with a tiny amount of custom C code to interface to the file system, the windowing and graphics innards, the IO system, and the signalling / interrupt system.

The compiler was the hard part.

The compiler worked in the usual way, and bottomed out in a set of 16 primitives, called "primops" for "primitive operations." These primops would be compiled into machine instructions that were turned into bits by a program called the Lisp Assembly Program or LAP. LAP was small and also broken into machine dependent and independent parts.

There were also a set of source-to-source transformations specific to the platform as well as peephole type transformations for the final assembly language.

The compiler was parameterized by these primops and transformations. Once they were defined, the compiler would compile the entire Lisp system—which was entirely written in Lisp—into a pile of compiled files and a boot file.

43. /SWITCH/

This approach enabled us to write code that looked like this.

CAR takes a CONS cell—a pair—as an argument and returns the first element of the pair. From CONS cells, lists can defined. Looking at this definition you might wonder how it terminates.

The trick is that the compiler "open codes" the call to CAR inside. That is, the compiler knows that the source-code call to CAR really means a call to the primop called %CAR, which is then translated into assembly language by the compiler, and the result is that a definition like this provides library-like access to the primop for %CAR.

44. /SWITCH/

The boot code and the platform-dependent interface code were then combined into a boot image by a mostly platform independent program we called the "Cold Loader."

The boot image contained a Lisp function called "fasload" which loaded the rest of the compiled code. Fasload could also execute "toplevel forms" during loading, so that the system could self-initialize based on environmental conditions. The result was a running Lucid Common Lisp.

We had extensive test suites, and these would be run automatically.

45. /SWITCH/

The story's not quite over for booting up. A development machine was used for the initial compilation and to prepare things up through the boot image. That's the green machine on the left. Then the boot image read in the compiled files and initialized the system while running on the target machine—the red one on the right.

Experience taught us that some primop or platform-dependent transformation bugs would show up only in a third generation system, and so we would then repeat the compile-load-test cycle twice more while running entirely on the target machine.

46. /SWITCH/

All this technology was dynamic. You could load the descriptions of any number of target machines (here there are 3), and dynamically reconfigure the compiler to produce code for them. This made it easy to make patches for a bunch of platforms at the same time.

47. /SWITCH/

The target machine model included stuff like how stacks worked, how data was moved around, what kinds of registers there were, how variables and structures were accessed, what side effects different things had,

48. /SWITCH/

how runtime tags were represented and interpreted, how stack frames were laid out, and how primops were expanded into machine code.

49. /SWITCH/

These platform-dependent things were collected into objects called "Compiler Machines," which also included constants, compiler macros for the target machine, the primops, opcode macros, operand decoders, register descriptors, and LAP methods for abbreviating assembly language statements.

50. /SWITCH/

And even though in 1984 object-oriented programming was not quite in vogue, we used the idea to put these compiler machines into inheritance hierarchies.

51. /SWITCH/

How did this all pan out? The entire Lisp system was about half a million lines of code. 4% was machine dependent. Porting took about a month with a viable boot image ready to try in a week or two.

65% of the compiler was machine independent, with 5% of the machine dependent code going to tagging and 30% for CPU dependencies.

We had the fastest running Lisp system on every platform we ran on, and because our test suite was strong, we were called the "Maytag of Lisp." We ran on workstations, minicomputers, IBM mainframes, 6800s, SPARCs, MIPS, Power PCs, Alphas, RT PCs, Primes, Apollos, Vaxes, and numerous versions of Unix, VMS, CMS, and the strange and wonderful Apollo operating system. We put every Lisp implementation out of business but two of them (there were about 15, including DEC, TI, Symbolics, LMI, Xerox, Three Rivers, Tektronix, and IBM).

Our code on benchmarks suitable to both were competitive with C.

52. /SWITCH/

Thus far what I've done is to muddy and complicate your ideas about what science and art are. I've deliberately made science look confusing, and in particular made it appear to be seeking knowledge without a solid philosophical footing. And I've made art look more principled than the simple yowling into the night many take it to be. That is, I've tried to explode your stereotypes.

My story about Lucid and its simple product-line approach to portability was meant to show you how ideas you're deeply knowledgeable about today started out as sort of inept fumbling in the absence of science, engineering, or even vocabulary. All I wanted you to see were the outlines of your discipline faintly visible, perhaps as through mist, and that as simple practitioners we were simply trying to get work done while working with the stuff right in front of us.

The rest of this talk is about my discovery that art-science is just one among perhaps many dimensions that might describe how people work on creative endeavors. While preparing this talk, I found one particular dimension I had previously collapsed into the art-science one.

This will bring us back to noticing. How hard it is to *notice* when our rational minds try to layer our concepts on top of what is really out there. Even the already-crazy artists sometimes need to use artificial means to see things as they are.

53. /SWITCH/

Over the last 2000 years there have been about 900 attempts to invent a language that is better than natural language—either to be more logical, more precise,

54. /SWITCH/

be easier to learn, or to eliminate perceived design flaws. One question about natural languages is that even though people quite obviously made them, how were they designed? Were they designed in any sense? Do they reflect a certain structure in our brains, our DNA, the background order of the universe?

Languages considered as engineering works are obviously flawed, and the good scientist or engineer desires to do better.

55. /SWITCH/

But natural languages have a distinct advantage—they support discovery and thought formulation—of just the sort the artists we've looked at so far have talked about. The messiness of natural languages gives them a power that we miss when we write in artificial languages, such as programming languages.

56. /SWITCH/

And in the hands of a careful writer, natural language can be as precise as we need it to be. Most programming language specifications have a core in natural language.

Let's look at some code!

57. /SWITCH/

Beautiful. We all know a language like this one, so it should be clear what it does, no?

58. /SWITCH/

This code is almost the same as the previous, but its author did something different and additional. Art was added. Compilers don't care what names things have as long as they are precise and accurate in their distinctions. People are more interested in understanding, and here what can only be called art has added that aspect.

The messiness is missing, and so people naturally add it back by creating their own natural language on top of the parts of the artificial one that provide wiggle room. This is what's happened in this example.

59. /SWITCH/

Art can go pretty far in this direction. There is no need to run this shell script to get its humor. By itself it's a form of light verse, worthy of Ogden Nash, perhaps.

60. /SWITCH/

Earlier I talked about artistic processes. Poets and others sometimes engage in processes of defamiliarization in order to see the world differently. Here I used Babelfish to take a familiar poem and run it through a number of cycles of translation to and through other languages and back to English to undo the idioms and figures of speech that Frost used originally.

61. /SWITCH/

Just glance at it for a minute, and I'll read a bit of it to you soon.

62. /SWITCH/

This is Frost's original. Given the defamiliarization, the poet must make something of it. Look at the second and third stanzas **/SWITCH/** while I read the distorted version.

His harness bells are his only user interface.
These bells are installed to a flange by some wiring, and so
he gives the flange a shock, vibrating the wires,
thereby jolting the bells (giving them a restlessness)
in order to pose me a question:
Is there some kind of mistake here?
Surely a certain error exists.
He is a small horse.

Some poetry is supposed to be funny.

63. /SWITCH/

This is flarf, which is a type of poetry that uses as a first draft the search synopses returned by Google when given a couple of search words that don't usually go together. This is a process of man-computer exploration and discovery.

64. /SWITCH/

For 40 years Harold Cohen has worked on a computer program called Aaron that produces art by itself. This is one of its paintings. People collect them. The spectrum of artificial assistance in art (and in science) is quite broad.

65. /SWITCH/

While studying the spectrum of scientific to artist creation I stumbled across another dimension that muddies things. This is the dimension of experimental to conceptual.

66. /SWITCH/

When I first started thinking about art, I believed that all art—especially writing—was what is called experimental art. I believed that a good writer just started out in some direction—even if he or she planned out the piece in great detail—but sooner or later the piece would take over and the writer would be along for the ride.

67. /SWITCH/

But this isn't true. The poems I just showed you, and to an extent the code as well, are not the product of unbounded exploration and discovery. Each of them was based on a concept.

68. /SWITCH/

In the defamiliarized Frost poem, my concept was to use Babelfish to make a set of ridiculous translations. I had it translate the poem from English to Greek, then to Korean, then to Japanese, then to German, and back to English; I used a set of cyclic translations like that as the starting point for my poem. This is conceptual art. So is flarf.

69. /SWITCH/

Most of my poems are experimental in the sense defined here. I like the subject to emerge, and then I study it and revise it into a poem.

I find conceptual poems easier to write, and in my practice of writing a poem a day, when I'm stuck, that's what I fall back on.

70. /SWITCH/

In painting, Cézanne is an experimental artist and Picasso is conceptual. Generally the masterpieces of experimental artists are done at an older age than for conceptual artists.

71. /SWITCH/

Back in the 1970s, a curmudgeonly professor at the MIT AI Lab noticed the same thing, and though I knew him and read his book, I never made the connection. Joseph Weizenbaum is best known for writing the *Eliza* program, which pretended—in a trivial fashion—to be a Freudian therapist.

72. /SWITCH/

But he was irked by the programmers at the AI Lab, especially Richard Stallman. He painted a picture of the "ordinary professional programmer" and contrasted it with the "hacker," meaning someone like Stallman.

The professional programmer worked like a conceptual artist, making plans in detail, and perhaps not even executing them himself, but delegating that to an underling—or perhaps an apprentice.

73. /SWITCH/

The hacker—whom he delicately called a "compulsive programmer" —acts like an experimental artist, just digging in with only the dimmest glimmer of what to do, and eventually something comes out.

74. /SWITCH/

And like the experimental artist or poet, if anything good emerges it's because of technique and skill, not knowledge. Not something to form theories about, not something amenable to science.

75. /SWITCH/

Now I believe the space of how people work at creative endeavors is at least as complex as this.

Let's start with conceptual artists. Sometimes they are forced into it because their medium is not malleable, not like words on a page or in a file. (Note that the advent of computers for writing made it easier to be an experimental writer for having a much more malleable medium for the words to live on or in.)

Experimental artists are like Cézanne, like a lot of pre-computer-age poets, like hackers in the 1970s perhaps.

Conceptual scientists are like mathematicians and Einstein.

Experimental scientists are like engineers, or the complexity scientists who use agent-based simulations to understand systems and try to manipulate them.

Conceptual creatives like abstract things, like symbol systems or sketches that show only the big ideas; experimental ones like to get their hands or minds around the stuff itself. Kent Beck calls source code "stuff." Drew McDermott, a computer scientist at Yale, once said, "What I like about Lisp is that you can feel the bits between your toes."

Artists believe that the key part of making art is exploration and discovery, while scientists believe it's understanding and verification. Notice that I said "believe."

In fact, artists evaluate their discoveries and revise like crazy—writers convene circles of friends and workshops to examine and critique drafts. And in fact, scientists have nothing to understand and verify unless they have explored and discovered.

Scientists and artists are similarly blind to what they actually do and how they really work. The problem with this picture is that it is static—

76. /SWITCH/

—in actuality, everyone working on a creative project moves all over this 2-dimensional space while working. This is because they are all people, and people mostly work the same ways. Sometimes they explore and discover, sometimes they verify and control, sometimes they work in the realm of ideas, and sometimes they manipulate materials.

77. /SWITCH/

I've found that people have a hard time observing how they and other people actually work. Take Malcolm Sparrow, who gave a lunch keynote at an NSA workshop I was at not long ago. He started it by talking about how adults untie knots—and his recent book uses a knot for its cover art. They do it, he says, by carefully studying the knot, understanding how it works, and then formulating a detailed plan. That is, he says adults act like conceptual scientists.

78. /SWITCH/

Children, on the other hand, act like experimental artists. They immediately (or rather quickly) jump into the task. They pull and tug. Like children, they play. When I heard this, I immediately decided to doubt everything else he said.

79. /SWITCH/

Later research revealed that adults untie knots mostly the way Sparrow says children (foolishly) do. There are simple heuristics for things to look for, but when that fails, twist and push in a playful manner until the problem is easy.

80. /SWITCH/

Some advice I found talks about "playing" with the knot to explore it in hopes of discovering an easy solution. You use your senses to see and feel where things are loosening. Then it's persistent play. Failures and mistakes are part of the normal process.

81. /SWITCH/

Cézanne played. He struggled to develop an authentic observation of the seen world by the most accurate method of representing it in paint that he could find. To this end, he structurally ordered whatever he perceived into simple forms and color planes. His artistic goals were things he could only approach and never achieve,

82. /SWITCH/

and that's why he kept trying, with his best works coming late in life. His ideas were not concepts that came in a flash, but something he painted and painted and painted. He stalked art, he pursued it like a series pursues its convergence.

83. /SWITCH/

Picasso thought of himself as an adult in the Malcolm Sparrow sense. This is considered by many to be his masterpiece. He carefully planned it.

The painting portrays five nude prostitutes in a brothel in Barcelona. The figures are physically jarring, none conventionally feminine, all slightly menacing, and each is rendered with an angular and disjointed body shape. Two of the women have African-mask-like faces, giving them a savage and mysterious aura. This is a variant of Primitivism. Picasso also abandons perspective in favor of a flat, two-dimensional picture plane.

84. /SWITCH/

Picasso denied obvious connections to African tribal masks and the primitive painting of other cultures.

Picasso states that his art is the result of what he has found—as opposed to showing his seeking, as Cézanne did—but he claimed to not believe in research.

85. /SWITCH/

Yet, for the painting we just saw, Picasso did over 400 studies and sketches, which is a record. If this isn't playing, experimenting, and researching, I confess to not knowing what those things are.

Just a question: how well do you believe theoreticians and practitioners in your discipline observe what really goes on? Or are their observations colored by previously held concepts about the nature of what could or should be done?

86. /SWITCH/

Over the past century or so, there have been persistent and dedicated efforts to "scientifically" control development and production processes.

Such efforts have made a big difference to the quality of manufacturing though I can't say it's helped the quality of design much.

87. /SWITCH/

Scientific processes have also been pushed in the realm of software development, and I believe this is a significant error—an error based on misperceiving how work actually takes place in designing and creating software.

88. /SWITCH/

Many aspects of software development require an artistic approach, an approach that acknowledges that for some things, a process of exploration and discovery is required. Sometimes these artistic processes are experimental—because source code and software are malleable and subject to "play"—but sometimes they are conceptual, achieved through blackboard designs, role playing, simulations, and modeling. Moving through the art/science/concept/experiment space is fractal.

Artistic approaches work best when requirements are sparse, incomplete, or even simply wrong or poorly stated. They also work where invention is desired.

89. /SWITCH/

Yet, pressure continues to be placed on software creation to fall in line with scientific processes. This is because, I believe, people would really like to see software creation be an engineering discipline. Software creation is not there yet. And as we've seen, the traditional, naïve view of engineering is that it is the handmaiden of science.

90. /SWITCH/

The manufacturing metaphor came to be when producing code a line at a time looked a lot like laborers laying sleepers and rails.

But even in the production of software in situations where everything is well understood, mistakes are made and programmers spend their time trying to wrangle the code into behaving the way they clearly see it in their minds.

91. /SWITCH/

Always, the creation of software requires a cycle through the explore-discover-understand space. When we bother to watch how the work gets done—regardless of the theory that sits in your head about how it should be done—we see this everywhere in software development.

92. /SWITCH/

And this isn't only a local phenomenon—it happens at all levels, and in the higher levels it's called "refactoring." This is what Weizenbaum missed when he watched ordinary professional programmers—even though they didn't sully their hands with actually punching in code and overseeing its execution, they were forced to re-consider, replan, and reformulate, just as Picasso did in his sketchbooks.

93. /SWITCH/

People who dig into the nature of software creation recognize that in many cases there is a cycle of exploration and discovery with a heavy dose of getting it to work given the prehistoric tools we work with. This is in part because when we develop software we are creating novel worlds without imposed constraints on what's possible—there are only the constraints we imagine up to put in place.

94. /SWITCH/

We need to look closely at what creative work is really like and adopt techniques and processes that support it. Education is part of it. I've been trained as a scientist and as an artist, and I have to say that I can't report that one education was easier than the other, and I can't declare that the two halves of me sit in separate containers with only my name and body common to both—I use everything I've learned for everything I do.

Possibly artistry has been pushed aside because it's not well understood. A common definition of an artist is "a person who expresses himself through a medium." Certainly some artists do this, but not all, and perhaps not the majority. It would be just as true to say that an engineer is "a person who expresses himself through bridges."

I believe everyone works in wandering curves in the 2-dimensional space of the art-science / concept-experiment spectra. We need to notice that and support it.

Explore.

Discover.

Understand.